Online Learning and Control of Dynamical Systems from Sensory Input

Oumayma Bounou, Jean Ponce, Justin Carpentier

Online Learning and Control of Dynamical Systems from Sensory Input

Oumayma Bounou[1], **Jean Ponce**[1, 2], **and Justin Carpentier**[1]

[1]Inria and Département d'Informatique de l'Ecole Normale Supérieure, PSL Research University
[2]Center for Data Science, New York University

Abstract

Identifying an effective model of a dynamical system from sensory data and using it for future state prediction and control is challenging. Recent data-driven algorithms based on Koopman theory are a promising approach to this problem, but they typically never update the model once it has been identified from a relatively small set of observations, thus making long-term prediction and control difficult for realistic systems, in robotics or fluid mechanics for example. This paper introduces a novel method for learning an embedding of the state space with linear dynamics from sensory data. Unlike previous approaches, the dynamics model can be updated online and thus easily applied to systems with non-linear dynamics in the original configuration space. The proposed approach is evaluated empirically on several classical dynamical systems and sensory modalities, with good performance on long-term prediction and control.

1 Introduction

1.1 Context and motivation

Providing complex machines such as robots with the capacity of learning effective models of their dynamics from their physical interactions with the world is a key enabling factor for deploying them outside of laboratories and factories. Analytical models of these interactions through state-based representations are traditionally derived from physical principles and used to solve challenging problems in various domains. They have been successfully deployed on heterogeneous systems such as robots or aircrafts/spacecrafts. Through the prism of control theory, these analytical models enable engineers to characterize and study the intrinsic dynamical properties of systems (contraction, stability, etc.), which are essential for system comprehension and for critical applications. In this context, optimal control theory [1] and its online counterpart known as model predictive control [2], exploit these models to plan and control complex dynamical systems (robots, rockets, etc.). Yet, the physical interactions of machines with the world are subject to complex natural phenomena, hardly describable by closed-form mathematical models.

By leveraging recent progress in machine learning, data-driven approaches [3, 4] have emerged as an effective way of learning advanced models of complex physical phenomena such as turbulence in fluid mechanics [3, 5, 6]. In particular, the Koopman formalism [7] is an attractive and versatile mathematical framework to learn and analyse complex dynamics. The overall objective of this formalism is to map the intrinsic state space of the system to a typically infinite-dimensional vector space where the system dynamics evolve linearly. However, exhibiting such a map remains challenging, especially in the context of instrumented systems such as robots, which evolve in the world by exploiting their sensory input. The sensory inputs used to build a representation of the internal state of the system and the world may be of various nature: images and force, temperature values or encoder readings,

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

etc. Importantly, to be effective for real-world applications such as the control of robotic systems, learned dynamical representations should also offer online update capacities, to account for new measurements as they come, which is essential to get long-term accurate prediction capacities in changing environments, and update the internal dynamics model accordingly.

1.2 Related work

Learning dynamics from data. Data-driven approaches based on the Koopman theory [7] and its approximation using the Dynamic Mode Decomposition (DMD) [3] have been used in the context of building dynamical models from measurements [6, 8, 9, 10, 11, 12, 13, 14, 15, 16]. A first group of approaches builds handcrafted functions to map from the measurements space to a latent space where the dynamics are forced to be linear [8, 9, 10, 11]. Such approaches are suitable when one specific system is being considered but cannot generalize to new systems, or to systems with varying physical properties. A second group of approaches learns a mapping from the measurements space to a latent space in the form of parametric functions. These approaches are more flexible, and showed promising results for both prediction [12, 13, 14, 15] and control [6, 16, 17]. [13], [14] and [15] learn an approximation of the DMD matrix in the form of a trainable linear layer. While this setting is satisfying when only a single dynamical system is considered, it becomes inappropriate to build more generic models since the same matrix can not be used to approximate the dynamics of different systems. To address this issue, [6, 12, 16] look for an approximation of the DMD matrix as a solution to a least-squares problem associated to a trajectory of a system. This enables identifying a linear model from a given trajectory of a system. In another line of work, [17] seek to identify a locally linear model of a system's dynamics in a learned representation space, through identifying a matrix to transition from one latent representation to the next one. However, their approach does not generalize to globally linear models.

Learning and control of forced dynamics. An extension of the DMD framework to actuated systems for model identification and control purposes has been introduced in [18], where the transition from one time step to the next is modeled by two matrices: the previous DMD matrix and a control matrix. Building upon this framework, [6, 14, 16] focused on learning an embedding for actuated systems. [6] and [14] parametrize the control matrix as a linear layer that is learned along with the embedding, while authors in [16] estimate the control matrix similarly to the DMD matrix (i.e.; from an input system trajectory representation), but assume they have access to the true state of the system, instead of only to measurements. [10, 11, 19, 20] also follow the same approach, although they do it on hand-crafted functions of either the measurements or the states, which they assume known in the case of [20]. In this work, we look for both the DMD and control matrices as solutions to a single least-squares problem including state representations and control. However, we assume the true state of the studied systems is unknown, and aim at leveraging sensory information to learn a linear representation of its dynamics. Contrary to the aforementioned approaches, we also introduce a proximal reformulation of the standard DMD approach based on a singular value decomposition (SVD) [21] to avoid introducing any extra hyperparameter of the rank truncation of the SVD that is required in the classical DMD setting. To account for changing dynamics, [20] estimate initial DMD and control matrices from an initial set of pairs of consecutive states of a given system, and perform active learning by updating their matrices with new executions of the system. In this work, we estimate matrices that are specific to a given trajectory, and update them with new measurements.

Future video prediction. The video prediction setting which is the focus of our paper has received a lot of attention in the past few years in computer vision, and various approaches for modeling the time evolution have been proposed. [22, 23, 24] do it through recurrent neural networks. Several approaches focused on predicting the next frame only conditioning on one previous frame [25, 26, 27, 28]. However, image data does not contain any velocity information. Indeed, if not explicitly modeled, this information can only be obtained when considering at least two consecutive frames, which is the approach we follow in our work. Some approaches [28, 29] model the multiplicity of future possible outcomes with variational approaches [30] however, in this work, we focus on the deterministic property of physical systems and only consider one possible future, like [31, 32, 33].

Figure 1: General overview of the proposed pipeline: the measurements $[d_1, \ldots, d_T]$ are first encoded with Φ_θ to codes $[z_1, \ldots, z_T]$. Only the m first codes are used to estimate the linear system dynamics A arising from the least-squares minimization problem (3). Using this linear dynamics A and the last code z_m, the last $T - m$ codes are predicted. The resulting reconstructed measurements $[\hat{d}_1, \ldots, \hat{d}_T]$ are obtained by decoding with Ψ_θ the embeddings of the actual measurements $[z_1, \ldots, z_m]$ and the predicted embeddings $[z_{m+1}, \ldots, z_T]$.

1.3 Contributions

We introduce an effective approach to learning a linear state representation directly from raw sensor input (e.g., images) in the form of an instance of a nonlinear latent embedding of a state space and a linear autoregressive model for that embedding [15]. Unlike classical approaches [6, 12, 13, 14, 15, 16, 34], once learned, our model of the system dynamics can be updated online to account for new measurements at very low cost. It also involves several key innovations: **(1)** Contrary to classical machine learning approaches to this problem [25, 29, 35], our approach is firmly grounded in (applied) Koopman theory [4, 7, 36] and, as demonstrated in Section 3.4, it readily generalizes from identifying the dynamics of a system to actually controlling it. **(2)** Conversely, contrary to existing algorithms based on Koopman theory and dynamic mode decomposition [3, 37, 38], we show that online updates of the linear dynamics model can efficiently be implemented in our framework, leading to improved prediction results (Sections 3.2 and 3.3). **(3)** We introduce several technical improvements that make the proposed approach practical, including a block-companion matrix representation of the linear autoregressive model enabling the use of multiple frames to drive video synthesis (Section 2), and the use of a robust, parameter-free proximal iterative refinement algorithm [39] for the least-squares estimation of this model which proves crucial in practice in the context of a deep learning library such as PyTorch. Our last main contribution is (**4**) an extensive experimental evaluation of the proposed approach on multiple problems from control theory (Section 3.4). Our experiments include video simulations of complex dynamical systems for which we can provide accurate long-term predictions and that we can effectively control.

2 Online learning of system representations

Proposed approach: We propose to learn the dynamics of a complex physical system from sensory inputs using a linear auto-regressive model for a nonlinear latent embedding of the corresponding state space [4, 6, 10]. Concretely, we consider a discrete-time dynamical system defined by some transition function $L : \mathcal{X} \to \mathcal{X}$ over some (arbitrary) state space $\mathcal{X}$. In our setting, L is unknown, but its effect can be in fact (partially) observed through measurements (elements of $\mathbb{R}^p$, video frames in our case) acquired in successive states along trajectories of the dynamics. Our aim is to learn from such measurements (1) an implicit embedding $\varphi : \mathcal{X} \to \mathcal{Z}$ of the state space into some latent vector space $\mathcal{Z}$ using a parametric encoder $\Phi : \mathbb{R}^p \to \mathcal{Z}$ of the corresponding data, and (2) a linear model $A : \mathcal{Z} \to \mathcal{Z}$ of the dynamics in the latent space so that, for any x in $\mathcal{X}$, $\varphi(Lx) = A\varphi(x)$. In this setting, $h = \Phi(d)$ is the feature $\varphi(x)$ associated with the (unknown) state x in which d has been measured. There is no reason to assume the dynamics L to be linear (which may be meaningless anyway since $\mathcal{X}$ may not be a vector space, or more generally be endowed with a well-defined algebraic structure). However, we know from Koopman theory [7] that, when $\mathcal{Z}$ is taken to be the set of all real (or vector-valued) functions over $\mathcal{X}$, there exists a linear map $P : \mathcal{Z} \to \mathcal{Z}$, called the Perron-Frobenius operator [4, 40] such that $\varphi(Lx) = P\varphi(x)$ for any x in X. Since $\mathcal{Z}$ is typically of infinite dimension in this case, P does not normally admit a finite representation in the form of a matrix. In the practical case where the latent feature space $\mathcal{Z}$ is by design finite-dimensional, on the other hand, P is not guaranteed to be linear but, given m features $z_1, \ldots, z_m$ of dimension n, one can approximate its action using an $n \times n$ matrix A minimizing the loss:

$$\|[z_2, \ldots, z_m] - A[z_1, \ldots, z_{m-1}]\|_F^2. \tag{1}$$

This approach is called Dynamic Mode Decomposition (DMD) [3] and, together with its extensions [18, 37, 41], it has been successfully used to study many dynamical systems. In our setting, the features are given by $z_i = \Phi(d_i)$ $(i = 1, \dots, m)$ and correspond to m successive measurements, gathered in our case from some video clip. DMD typically operates on raw measurement data instead of a latent representation thereof. Learning the embedding function Φ requires some sort of supervisory information, and we follow the popular encoder/decoder framework [42] by also estimating the parameters of a decoding function $\Psi : \mathcal{Z} \to \mathbb{R}^d$ such that Φ and Ψ jointly minimize the regularized mean of the reconstruction error $\|d - \Psi \circ \Phi(d)\|_2^2$ over some sample of measurements. The full pipeline is composed of a learnable encoder/decoder module mapping from the measurement space to a latent state space, combined with a linear state representation which is learned from a sequence of embedded measurements (see Fig. 1). We now detail its different components.

Measurement embedding. The goal of the autoencoder is to learn an embedding of the measurements into a latent space representation of the state space. While input measurements may be high-dimensional objects (e.g. images), autoencoders are often capable of learning effective yet compact representations of the corresponding state space. Given a sequence of T measurements $[d_{1:T}]$, we are looking for two parametric functions: an encoder Φ_θ and a decoder Ψ_μ parametrized by $\theta \in \mathbb{R}^{n_\theta}$ and $\mu \in \mathbb{R}^{n_\mu}$, following the relation:

$$\Phi_\theta(d_t) = z_t \text{ and } \Psi_\mu(z_t) = \hat{d}_t \text{ for all t.} \tag{2}$$

Dynamic mode decomposition solves the least-squares problem associated with (1), which can be reformulated as:

$$\arg\min_A \frac{1}{2}\|Z_2 - AZ_1\|_F^2 \text{ where } Z_1 = [z_1, \dots, z_{m-1}] \text{ and } Z_2 = [z_2, \dots, z_m]. \tag{3}$$

A standard approach to solve this problem makes use of the singular value decomposition of the matrix Z_1 containing the collection of consecutive measurements, following the relation:

$$A^*(z_1, \dots, z_m) = Z_2 Z_1^+, \tag{4}$$

where Z_1^+ is the pseudo-inverse of Z_1. As in practice Z_1 is a singular matrix, ones needs to carefully choose a given singular-value threshold, either to damp the solution or to cut the spectrum of Z_1. Both solutions only provide approximate solutions of the original least-squares problem (3). Additionally, the SVD is differentiable only when all the singular values are different [43], which may not be the case in general and could lead to unstable behavior when performing (stochastic) gradient descent throw computational layers which include SVDs. To overcome this issue, we propose to rely on an alternative approach called proximal method of multipliers [39]. Instead of solving a classical least-squares problem, we suggest solving an equality-constrained quadratic programming of the form:

$$\min_{A \in \mathbb{R}^{n_z \times n_z}} \frac{1}{2}\|A\|_F^2 \text{ s.t. } Z_2 = AZ_1. \tag{5}$$

The proximal method of multipliers [39] augments the Lagrangian function associated to (5) with a proximal term over the dual variables associated to the constraint $Z_2 = AZ_1$, leading to:

$$L_\rho(A, \Lambda, \Lambda^-) = \frac{1}{2}\|A\|_F^2 + \sum_{t=1}^{T-1} \lambda_i^T (z_{t+1} - Az_t) - \frac{\rho}{2}\|\lambda_t - \lambda_t^-\|_2^2, \tag{6}$$

where ρ is the smoothing parameter over the dual, λ_t are the multipliers associated to the constraints $z_{t+1} - Az_t$, and λ_t^- is an estimate of the multiplier λ_t. Such saddle-point reformulation can be iteratively solved through the the Karush-Kuhn-Tucker (KKT) system of equations:

$$\begin{bmatrix} I_{n_z} & Z_1 \\ Z_1^T & -\rho I_{T-1} \end{bmatrix} \begin{bmatrix} A^k \\ \Lambda^k \end{bmatrix} = \begin{bmatrix} 0 \\ Z_2 - \rho\Lambda^{k-1} \end{bmatrix}, \tag{7}$$

where I_{n_z} corresponds to the identity matrix of dimension n_z and Λ^k is the stack of multipliers associated to each constraint $z_{t+1} = Az_t$. (7) is solved by iterative refinement until convergence to the fixed point solution. Typically, for small values of ρ (e.g. 10^{-8}), less than a dozen of iterations are needed to converge to the optimal solution and the approch converges for any value of ρ. The system of equations (7) is always invertible thanks to the lower right block $-\rho I_{T-1}$ and has a

condition number closed to the one of Z_1. From a computational perspective, such formulation can be efficiently solved by exploiting the inherent sparsity of the 2x2 block matrix, using a Cholesky decomposition of the KKT matrix. Contrary to SVD, this approach can be easily differentiated by backward propagation of the gradient over the iterative process, which appears as an appealing feature for differentiable programming [44] in the context of deep learning.

Online dynamic mode decomposition for long-term prediction. Once built from a sequence of codes obtained from initial measurements, the DMD matrix A can be used to predict the codes of future states given a known initial code. As classically done in [6, 38, 15], only the first m measurements are used to estimate A, and future codes are predicted as:

$$\hat{z}_{m+t} = A^t z_m \;\text{ for } t \geq 1 \text{ s.t. } A^t \overset{\text{def}}{=} \underbrace{A \times \cdots \times A}_{\text{t times}}. \tag{8}$$

In this framework, the matrix A is never updated, thus never using potential new measurements. For long sequences, this might lead to inaccurate predictions as A remains static and is only estimated from codes of the first m measurements. While this is enough for simple systems (e.g, a pendulum), it becomes a limitation when more complex dynamical systems are considered (e.g, a cartpole). To overcome this issue, we propose to update A when new measurements are acquired, making the learned representation adaptive. We account for this during training by artificially extending the initial sequence used to compute A with additional latent codes of measurements. Re-starting prediction from codes of the new measurement should force the prediction error to diverge less quickly. We refer to the supplementary material for more details.

Capturing dynamics effects. Until now, we have considered one-step linear dynamics: each encoding z_t only depends on the state z_{t-1} linearly. However, most of the times, a measurement at a given time is only a partial observation of the state of the system and does not contain enough information to describe the full state of the system. For instance an image does not contain velocity information and at least two consecutive frames are needed to estimate it. To address this issue, we adopt a similar approach as proposed in [12] and consider a code z_t which linearly depends on a history of h previous codes $z_t^{:h} = [z_{t-h+1}, \ldots, z_t]$, such that:

$$z_{t+1} = A_1 z_{t-h+1} + A_2 z_{t-h+2} + \cdots + A_h z_t = A^{:h} z_t^{:h}, \tag{9}$$

where $A_i \in \mathbb{R}^{n_z \times n_z}$ and $A^{:h} = [A_1, \ldots, A_h]$. Like for Eq. (1), $A^{:h}$ can be obtained by solving an augmented linear least-squares problems, following the same proximal approach as before. Note that this formulation is equivalent to the one in Eq. (1), only augmented states $[z_t, \ldots, z_{t_{m-h+1}}]$ for $1 \leq t \leq m - h + 1$ would be considered, and the transition matrix would have a larger dimension of $hn \times hn$, depicting a block-companion structure of the form:

$$\tilde{A} = \begin{bmatrix} 0 & I & 0 & \ldots & 0 \\ 0 & 0 & I & 0 & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & 0 & I \\ A_1 & A_2 & \ldots & \ldots & A_h \end{bmatrix} \;\text{ s.t. }\; z_{t+1}^{:h} = \tilde{A} z_t^{:h}. \tag{10}$$

Such a scheme is classical when building models of dynamical systems from measurements and we refer to [45] for a comprehensive view.

Learning forced dynamics for control. Let us now consider an actuated system for which we have a sequence of measurements $d_{1:T}$. We assume the system was actuated with a sequence of control inputs $u_{1:T-1}$ that we have access to. The goal is to learn a representation space of the states of the system, from the measurements $[d_{1:T}]$ and the control inputs $[u_{1:T}]$ such that in this representation space, the evolution is linear on both states and control inputs. Following the extension of the DMD approach to control [18], we look for a latent space and matrices $A_1, \ldots, A_h$ and B such that we have for all t:

$$z_{t+1} = A^{:h} z_t^{:h} + B u_t. \tag{11}$$

Unlike [6], we do not treat B as a learned parameter. We also perform least-squares regression to find $A^{:h}$ and B following the same approach as before. Once these matrices are identified, they can be exploited within the standard linear quadratic regulator setting [1] for control. This is possible because we make the assumption that B is independent of z. In the case where this assumption does not hold, one can refer to the iterative linear quadratic regulator setting [46].

Loss function. To learn the parameters associated to our pipeline, we minimize the empirical risk of the L_2 reconstruction loss composed of two main terms: a first term associated to the autoencoder and a second term accounting for the desired linear prediction in the state space. The full loss is given by:

$$\mathcal{L}_{\theta,\mu}(\{d_{1:T}\}_{i=1,\dots,N}) = \frac{1}{N}\sum_{i=1}^{N}\underbrace{\sum_{t=1}^{m}\|d_t^i - \Psi_\mu(\Phi_\theta(d_t^i))\|_2^2}_{\text{Auto-encoder loss}} + \underbrace{\sum_{t=m+1}^{T}\|d_t^i - \Psi_\mu(A_i^{t-m}\Phi_\theta(d_m^i))\|_2^2}_{\text{Prediction loss}}. \tag{12}$$

To simplify the notations, we do not account for the terms associated to the history ($A^{:h}$ and $z_t^{:h}$). In the online setting, predictions in the latent space following a measurement d_j obtained at time $j > m$ are performed as $A^{t-j}\Phi_\theta(d_j)$ for $t > j$.

3 Results

We apply our learnable framework of systems dynamics from sensory input on challenging tasks. In particular, we address the tasks of future state prediction and control from raw video data, including both unactuated (unforced dynamics) and actuated (forced dynamics) systems.

3.1 Experimental setup

Datasets. We generate three black-and-white 64×64 video datasets of classical dynamical systems using the Pinocchio library [47] for the simulation and Panda3D-viewer [48] for the rendering: a simple pendulum, a double pendulum[1] and a cartpole. The cartpole we consider in our experiments is composed of a horizontal cart that is allowed one translation over an axis, to which a pole is attached and is allowed a rotation over an orthogonal axis. For each of these systems, we consider both unactuated and actuated versions, and generate training datasets composed of 4000 trajectories of 5 seconds ($T = 100$) in the first case, and 10 seconds ($T = 200$) in the second, with measurements every 50 ms. In the case of unactuated systems, we consider systems with varying physical parameters (masses, lengths) while in the case of actuated systems, we consider systems with the same physical parameters. In both cases, trajectories are generated with different initial conditions (angular positions and velocities). More details on the datasets generation can be found in the supplementary material.

Architecture. The only learnable parameters in our model are those of the autoencoder. The encoder is made of 6 blocks of 3×3 convolutions followed by max-pooling, batch normalization and ReLu layers, except for the last block which does not have a ReLu layer. The decoder is a symmetric copy of the encoder. The dimension of the codes in latent space corresponds to the bottleneck size of the autoencoder, set to $n_z = 8$ in all our experiments. More details on the architecture can be found in the supplementary material. Each learning problem takes about 3 hours to solve on an Nvidia RTX6000 GPU.

Evaluation. The models of the unactuated systems are trained on 5 s sequences and evaluated on 15 s sequences, and those of actuated systems were trained on 10 s sequences and evaluated on 20 s sequences. Longer sequences are used for training actuated systems since in this case we seek to identify both dynamics (A) and control (B) matrices in latent space. For all systems, we have evaluated the prediction ability of trained models with the average RMSE loss over time. In Figures 3, 4, 5, the first vertical dashed line indicates the number of time steps used to estimate the dynamics matrix (and the control matrix in the case of forced dynamics), and the second vertical dashed line indicates the duration for which the models were trained.

3.2 Long-term prediction of unforced dynamics

Figure 3 shows the prediction error over an horizon of 15s for the pendulum and cartpole systems. For both systems, the error over the first time steps before the first vertical dashed line is the autoencoder error. Starting from the first vertical dashed line, the error is the prediction error. The red curve

[1] illustrated in the supplementary material.

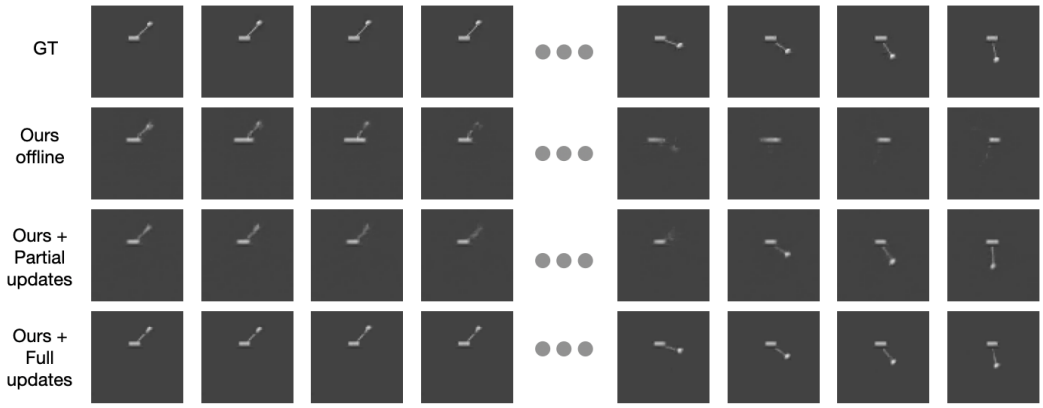

Figure 2: **Impact of online updates on the quality of the prediction for the cartpole.** The first row shows ground truth images. The second row shows predicted frames without updates. The third row shows predicted frames with our model trained with partial updates. The last row shows predicted frames with online updates performed at each new measurement.

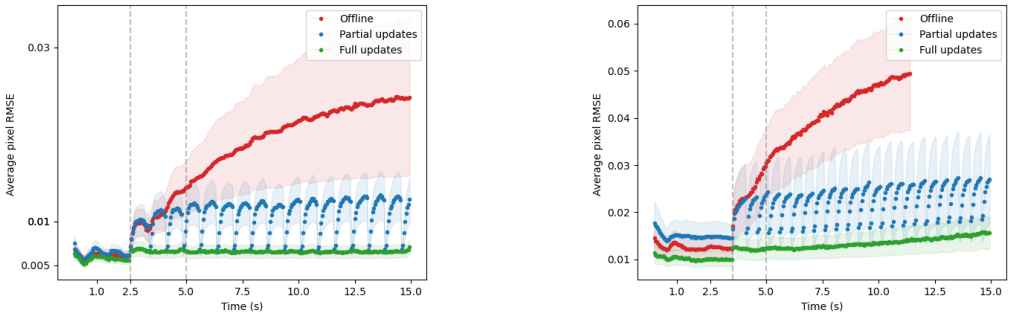

Figure 3: Average per-pixel RMSE loss over a 15s prediction horizon. See text for details. **Left:** pendulum. **Right:** cartpole.

corresponds to the case where the matrix A is computed using a fixed number of time steps and never updated. The blue curve corresponds to the case where the matrix A is first computed using a fixed number of time steps, then updated at fixed intervals (every 750 ms for the pendulum, and every 50 ms for the cartpole). The green curve corresponds to the case where the matrix A is updated every 50ms. This case corresponds to predicting only one step ahead, which is why the error is low and constant over time. For both systems, the prediction error grows with time when the matrix A is not updated, which demonstrates the necessity of online updates if one wants to perform accurate long-term prediction. The impact of the online updates on the quality of predicted frames of a cartpole can be seen in Fig. 2. The qualitative predictions are consistent with the evolution of the reconstruction error over time from Fig. 3 (right).

We compare our future prediction approach to the standard method of learning the matrix A as a parameter of the model as in [15]. We train and evaluate both approaches on two different datasets of pendulums, the first one containing trajectories of the same pendulum (mass $m = 1$kg, length $l = 0.6$m) with different initial conditions, and the second one containing trajectories of different pendulums (mass $m = 1$kg, length $0.3 \leq l \leq 0.8$). Figure 4 highlights the fact that computing A as the solution of a least-square problem (3) leads to more accurate predictions than learning with a fixed matrix A. In the supplementary material, we show that while the predictions are correct qualitatively on the first dataset even for a long horizon, they are wrong on the second dataset as not even the first predicted frame is correct. This is expected as a single matrix A cannot be used to model the dynamics of systems with different physical parameters.

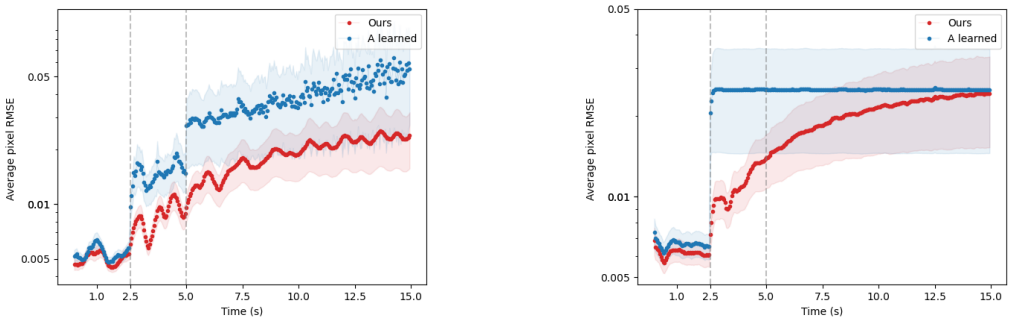

Figure 4: Average per-pixel RMSE loss over a 15s prediction horizon. **Left:** pendulum with length 0.6 m. **Right:** pendulums with lengths varying from 0.3 to 0.8 m.

3.3 Long-term prediction of forced dynamics

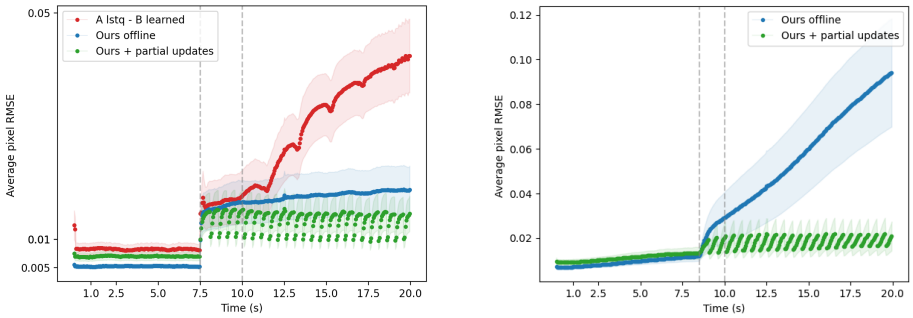

Figure 5: Average per-pixel RMSE loss over a 20s prediction horizon. **Left:** actuated pendulum. **Right:** actuated cartpole.

We consider the case of controlled pendulum and cartpole systems. Figure 5 shows the prediction error over a horizon of 20s of actuated pendulum and cartpole. The importance of (partial) online updates is demonstrated as without it the prediction error grows quickly over time.
We have compared our approach to the approach presented in [6], where the dynamics matrix A is the result of a least-square problem similar to ours, while B is a linear layer learned as a trainable parameter along with the autoencoder parameters. We adapt their approach to our setting in order to **(1)** include time-delay embeddings because our measurements are images, and **(2)** use an autoencoder tailored to our datasets, for fairer comparison. Even for offline prediction, our approach outperforms our implementation of [6] in the case of the actuated pendulum (Fig. 5, left), which validates the need for a learned specific control matrix B per trajectory. Having a constant B would imply that a given sequence of control inputs has the same effect on the state of the system no matter when along in the trajectory it is applied, presupposing a stationary regime of the states of the system, which is not the case for most real physical systems. This is all the more validated with the cartpole example in Fig. 5 (right), for which the approach with a static matrix B is unable to make accurate predictions in the 2.5s horizon, because of the chaotic regime of the system. More details on shorter-term prediction horizons of this approach can be found in the supplementary material.

3.4 Control from video inputs

We demonstrate the effectiveness of our approach on the control of simulated physical systems directly from visual inputs. We consider here the case of controlling a forced system such as a

pendulum. Such a system is highly non-linear and does not exhibit any linear state representation. Using our pipeline, we learn a linear-state representation (A, B) which can then be plugged into classic linear quadratic control settings [1] similarly to [49, 10, 16, 6].

Let us assume the system is at some initial configuration d_1 in the measurements space, and our goal is to drive it to a target configuration d_f given in the image space at time T_c. We are looking for a sequence of control inputs $[u_1, \dots, u_{T_c}]$ that minimizes:

$$\min_u \sum_{t=2}^{T_c} (z_t - z_f)^T Q (z_t - z_f) + \sum_{t=1}^{T_c - 1} u_t^T R u_t \quad \text{s.t.} \quad z_{t+1} = A z_t + B u_t \text{ and } z_1 = \Phi_\theta(d_1), \quad (13)$$

where Q and R are the standard symmetric positive definite matrices associated to the LQR setting.

Figure 6 shows the trajectory obtained for driving the pendulum from the initial positions (red) to a target configuration (green). As our representation is with delay embeddings with an history of 2 frames, we need to constraint the QP problems on two initial conditions. It is important to notice that we were not able to run this experiment with a single image history. Indeed, a pendulum and other mechanical systems are subject to gravity, which is a second-order information, which cannot be retrieved from a single image.

Figure 6: **Illustration of pendulum control.** Starting from two consecutive configurations (in red), control inputs are applied to the pendulum to force it to be in an inverted position in a horizon of 0.5 s. Final position is the target configuration (in green).

3.5 Limitations

In all the experiments, we use an embedding of the state of relatively small dimension $n_z = 8$ compared to the configuration space dimension of the studied dynamical systems. Yet, while we manually choose this value, the issue of choosing the correct dimension of linear state space representation remains open. Additionally, all our experiments were done on simulated and low-dimensional systems without considering noise on the sensory inputs. A next step to our approach would be to include real systems or measurements of real systems of higher dimension (e.g, real video datasets). Finally, our approach is motivated by the Koopman operator theory [7], through the DMD approximation [3]. While this approximation works in practice, no theoretical bound of this approximation that we are aware of has been established yet, and we believe that finding one remains an open problem. Some previous work ([41], [50]) discussed this issue from a spectral point of view by either showing that DMD (or EDMD) modes are a good approximation of the true Koopman eigenfunctions for specific problems when these are known analytically, or that they provide a correct parametrization of the system. Again, as far as we know, assessing the accuracy of the DMD/EDMD approximation to the Koopman operator for nonlinear dynamics remains an open problem.

4 Conclusion

We have introduced a trainable framework for learning an embedding of the state space of physical systems into a latent space where the dynamics are linear, directly from raw sensor inputs (images). Contrary to previous works, our computational framework allows learning system state representations

which are adapted to online updates, leading to more accurate predictions than offline models. This feature is also essential for online control of complex instrumented systems such as robots, where a compact state representation needs to be updated to account for dynamical changes in the environment. Additionally, our framework exhibits long-term prediction capabilities, an essential feature for planning and control. It is also able to learn complex embeddings for dynamical systems with varying parameters, which is an appealing feature for systems which operate in changing environmental conditions. We have applied it to future state prediction and control from input videos on several challenging dynamical systems. As future work, we plan to extend our approach to operate on a heterogeneous set of sensor inputs and to apply it on real robotic systems for solving fine manipulation tasks by directly exploiting vision and tactile feedback, similar to the experimental setting developed in [35]. We plan to extend our approach to 3D examples by accounting for multi-view measurements, in a similar spirit to what authors in [51] proposed.

Broader impact. This work is mostly a theoretical contribution to the online learning of complex dynamics from sensory input and its potential positive or negative impacts are similar to the ones of the field. Applications such as the learning of complex dynamics for machines interacting with the world or humans, among them manipulation or locomotion tasks in robotics, may lead to questionable use in surveillance or military contexts. Yet, we believe that developing generic approaches to learning complex dynamics with structured and interpretable components is part of a more general trend towards more structured learning algorithms. This could results in more explainable and efficient algorithms, a central topic for the ethical and ecological issues of AI.

5 Acknowledgements

We warmly thank Armand Jordana for fruitful discussions. This work was supported in part by the HPC resources from GENCI-IDRIS (Grand 2020-AD011011263R1), the Inria/NYU collaboration, the Louis Vuitton/ENS chair on artificial intelligence and the French government under management of Agence Nationale de la Recherche as part of the "Investissements d'avenir" program, reference ANR19-P3IA0001 (PRAIRIE 3IA Institute).

References

[1] D. Liberzon, *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press, 2011.

[2] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[3] P. Schmid, "Dynamic mode decomposition of numerical and experimental data," *Journal of Fluid Mechanics*, pp. 5–28, 2010.

[4] S. L. Brunton, M. Budišić, E. Kaiser, and J. N. Kutz, "Modern koopman theory for dynamical systems," *arXiv preprint arXiv:2102.12086*, 2021.

[5] K. Taira, S. L. Brunton, S. T. Dawson, C. W. Rowley, T. Colonius, B. J. McKeon, O. T. Schmidt, S. Gordeyev, V. Theofilis, and L. S. Ukeiley, "Modal analysis of fluid flows: An overview," *Aiaa Journal*, vol. 55, no. 12, pp. 4013–4041, 2017.

[6] J. Morton, A. Jameson, M. J. Kochenderfer, and F. D. Witherden, "Deep dynamical modeling and control of unsteady fluid flows," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada* (S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), pp. 9278–9288, 2018.

[7] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[8] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLOS ONE*, vol. 11, p. e0150171, Feb 2016.

[9] I. Abraham, G. de la Torre, and T. Murphey, "Model-based control using koopman operators," *Robotics: Science and Systems XIII*, Jul 2017.

[10] H. Arbabi, M. Korda, and I. Mezic, "A data-driven koopman model predictive control framework for nonlinear flows," 2018.

[11] D. Bruder, C. D. Remy, and R. Vasudevan, "Nonlinear system identification of soft robot dynamics using koopman operator theory," 2019.

[12] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning koopman invariant subspaces for dynamic mode decomposition," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[13] O. Azencot, N. B. Erichson, V. Lin, and M. Mahoney, "Forecasting sequential data using consistent koopman autoencoders," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 475–485, PMLR, 13–18 Jul 2020.

[14] Y. Xiao, X. Xu, and Q. Lin, "Cknet: A convolutional neural network based on koopman operator for modeling latent dynamics from pixels," 2021.

[15] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, Nov 2018.

[16] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba, "Learning compositional koopman operators for model-based control," 2020.

[17] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," *arXiv preprint arXiv:1506.07365*, 2015.

[18] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[19] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PloS one*, vol. 11, no. 2, p. e0150171, 2016.

[20] I. Abraham and T. D. Murphey, "Active learning of dynamics for data-driven control using koopman operators," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1071–1083, 2019.

[21] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, p. 127–239, Jan. 2014.

[22] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *International conference on machine learning*, pp. 843–852, PMLR, 2015.

[23] L. Castrejon, N. Ballas, and A. Courville, "Improved conditional vrnns for video prediction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7608–7617, 2019.

[24] R. Villegas, A. Pathak, H. Kannan, D. Erhan, Q. V. Le, and H. Lee, "High fidelity video prediction with large stochastic recurrent neural networks," *arXiv preprint arXiv:1911.01655*, 2019.

[25] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman, "Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks," *arXiv preprint arXiv:1607.02586*, 2016.

[26] W. Lotter, G. Kreiman, and D. Cox, "Deep predictive coding networks for video prediction and unsupervised learning," *arXiv preprint arXiv:1605.08104*, 2016.

[27] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, "Learning to generate long-term future via hierarchical prediction," in *international conference on machine learning*, pp. 3560–3569, PMLR, 2017.

[28] E. Denton and R. Fergus, "Stochastic video generation with a learned prior," in *International Conference on Machine Learning*, pp. 1174–1183, PMLR, 2018.

[29] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine, "Stochastic variational video prediction," in *ICLR*, 2018.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[31] C. Finn, I. Goodfellow, and S. Levine, "Unsupervised learning for physical interaction through video prediction," in *NeurIPS*, 2016.

[32] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *ICLR*, 2016.

[33] C. Vondrick, H. Pirsiavash, and A. Torralba, "Anticipating visual representations from unlabeled video," in *CVPR*, 2016.

[34] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *arXiv preprint arXiv:1312.0041*, 2013.

[35] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8943–8950, IEEE, 2019.

[36] M. Budišić, R. Mohr, and I. Mezić, "Applied koopmanism," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 4, p. 047510, 2012.

[37] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. Nathan Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, no. 2, p. 391–421, 2014.

[38] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, p. 1307–1346, Jun 2015.

[39] R. T. Rockafellar, "Augmented Lagrangians and applications of the proximal point algorithm in convex programming," *Mathematics of operations research*, vol. 1, no. 2, pp. 97–116, 1976.

[40] S. Klus, P. Koltai, and C. Schütte, "On the numerical approximation of the perron-frobenius and koopman operator," *arXiv preprint arXiv:1512.05997*, 2015.

[41] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data–driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[42] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length, and helmholtz free energy," *Advances in neural information processing systems*, vol. 6, pp. 3–10, 1994.

[43] M. B. Giles, "Collected matrix derivative results for forward and reverse mode algorithmic differentiation," in *Advances in Automatic Differentiation*, pp. 35–44, Springer, 2008.

[44] V. Roulet and Z. Harchaoui, "Differentiable programming à la moreau," *arXiv preprint arXiv:2012.15458*, 2020.

[45] L. Ljung, *System Identification*, pp. 163–173. Boston, MA: Birkhäuser Boston, 1998.

[46] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems.," in *ICINCO (1)*, pp. 222–229, Citeseer, 2004.

[47] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[48] I. Kalevatykh, "panda3d viewer." `https://github.com/ikalevatykh/panda3d_viewer`, 2019.

[49] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, p. 149–160, Jul 2018.

[50] H. Zhang, S. Dawson, C. W. Rowley, E. A. Deem, and L. N. Cattafesta, "Evaluating the accuracy of the dynamic mode decomposition," *arXiv preprint arXiv:1710.00745*, 2017.

[51] Y. Labbe, J. Carpentier, M. Aubry, and J. Sivic, "Cosypose: Consistent multi-view multi-object 6d pose estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[52] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification*, pp. 547–577, Springer, 2001.

[53] F. Witherden, A. Farrington, and P. Vincent, "Pyfr: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach," *Computer Physics Communications*, vol. 185, no. 11, pp. 3028–3040, 2014.

[54] Navier-Stokes equations, "Navier-stokes equations — Wikipedia, the free encyclopedia."

A Additional details on the experiments

A.1 Model architecture

The only learnable parameters (117,963 in all) in our model are those of the autoencoder. The encoder is made of 6 blocks of 3×3 convolutions with 16, 32, 64, 64, 32 and 8 channels followed by max-pooling, batch normalization and ReLu layers, except for the last block which does not have a ReLu layer. The decoder is a symmetric copy of the encoder. As our images are 64×64, the last convolutional block yields a feature map with 8 channels and 1×1 spatial dimension, which is reshaped into an 8×1 vector. The latent code we consider is thus directly the output of the convolutional encoder. Contrary to [6], we do not follow our encoder by fully-connected layers to obtain a compact code since the output of the convolutional encoder is alreay quite compact.
Models without updates take 2.5 hours to train on a Tesla V100-SXM2 GPU, and models with updates take 4 hours to train. Models including control take longer to train (4 hours without updates and 6 hours with partial online updates) since the video sequences considered are longer. All models are trained for 200 epochs with a batch size of 16 and a learning rate of 10^{-3} which is divided by 2 every 20 epochs.

A.2 Experiments on pendulum systems

A.2.1 Datasets generation

We have generated video datasets of cartpole and pendulum systems to which control inputs are applied. The length of the generated videos is $10\,$s and points of the system are generated every $5\,$ms, which is also the frequency at which controls are applied. Measurements (i.e, images) are taken every $50\,$ms. In the following, time steps will refer to measurements time steps (every $50\,$ms). To obtain videos of actuated systems, we have generated a set of reference trajectories and velocities to be followed by the systems. For simplicity, we specifiy the angle between the pole and the vertical, as well as its temporal derivative, and also control these two quantities. Thus in the case of the cartpole, only the pole is actuated, the translation of the cart remains free. Each reference trajectory is the sum of three sinusoidal signals with different frequencies. Starting from a random initial configuration, the system (pendulum or cartpole) receives a control input every $5\,$ms to match the target trajectory. The reference trajectories are of the shape:

$$\begin{cases} q_{ref}(t) = \sum_{i=1}^{3} q_{0,i} \sin(\omega_i * t + \varphi_i) \\ v_{ref}(t) = \sum_{i=1}^{3} q_{0,i} \omega_i \cos(\omega_i * t + \varphi_i), \end{cases} \tag{14}$$

where $q_{0,i}$ is the angular amplitude of the reference trajectory for the pole. In our experiments, $q_{0,i}$ was uniformly sampled between 0 and $\frac{\pi}{3}$ radians for the pendulum, and between 0 and $\frac{2\pi}{3}$ radians for the cartpole. We take $\omega_i = 2\pi f_i$ where f_i is uniformly sampled between 0 and 0.1 Hz for the cartpole system and between 0 and 0.3 Hz for the pendulum system. We take φ_i between 0 and 2π radians for both systems.
Having generated these reference trajectories, the control input to apply to the systems every $5\,$ms is determined as:

$$u(t) = -K_p(q(t) - q_{ref}(t)) - K_d(v(t) - v_{ref}(t)), \tag{15}$$

with $K_p = 100$ and $K_d = 10$.
Sinusoidal inputs and sums of sinusoidal inputs are commonly used to excite systems as they facilitate system identification [52]. This is why we choose to use reference trajectories in the form of a sum of sinusoids, since the controls we apply to the system are proportional to these trajectories, as can be seen in Eq. (15).

A.2.2 More results

Prediction quality. In all the following figures, the left block corresponds to the first predicted frames after those used to compute the matrix A, and the right block corresponds to predicted frames

after a horizon of 20 or 30 time steps. Figure 7 shows that for a simple system such as a pendulum with low amplitude oscillations (between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians), our offline model without any update is sufficient to predict future frames correctly. However, for more complex systems such as the pendulum with high amplitude oscillations (up to 2π radians) (Fig. 10, second row) or the double pendulum (Fig. 11, second row), our offline model yields blurry predictions, that can be corrected by performing online updates (even if they are not performed at every new measurement, but only every 15 measurements) (Figures 10 and 11, third row). Finally, for both systems, performing online updates at each new measurement yields visually perfect predictions at all time steps. (Figures 10 and 11, fourth row). The quality of the predictions for the double pendulum and the pendulum with high oscillations amplitude is consistent with the time evolution of the RMSE loss for these systems (Fig. 12), whose values are most likely over optimistic since most pixel values are 0 in our datasets. Figures 8 and 9 compare the quality of the predictions of our model to the quality of prediction of the baseline where the matrix A is learned as an additional parameter, and is thus constant over all the dataset. In this case, the matrix A is not computed using codes of past frames, it is instead learned, along with the parameters of the autoencoder. There is thus one single matrix A that is used for the prediction of future frames of different trajectories. Figure 8 shows that when models are trained on a dataset of a single pendulum with different trajectories (different initial conditions: initial position and velocity), the baseline gives good short-horizon predictions (left block) but poor predictions for longer horizons (right block). Our model does not exhibit such limitations. Figure 9 shows how the baseline model is unable to predict future frames correctly, for even a single step in the future (first frame of the left block), when it is trained on a dataset with multiple pendulums. This is not surprising as a single matrix A can not account for the dynamics of several different systems.

Figure 7: **Prediction for the pendulum with low oscillations amplitude (between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians) on a dataset of pendulums of length varying between 0.3 m and 0.8 m.** The first row shows ground truth (GT) images. The second row shows predicted frames with our model without updates. In the case of this simple system, our model without updates is enough.

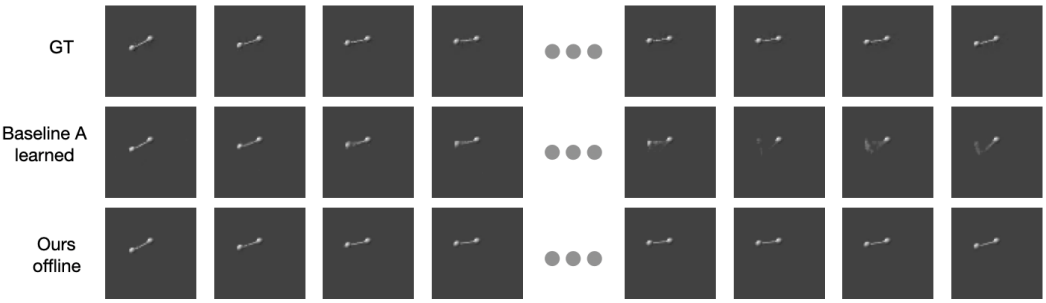

Figure 8: **Prediction for the pendulum with low oscillations amplitude (between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians) and length 0.6 m.** The first row shows ground truth (GT) images. The second row shows predicted frames without updates with the baseline model where the matrix A is a learned parameter. The third row is our model without updates.

Figure 9: **Prediction for the pendulum with low oscillations amplitude (between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians) with lengths varying from 0.3 to 0.8 m.** The first row shows ground truth (GT) images. The second row shows predicted frames with the baseline model where the matrix A is a learned parameter.

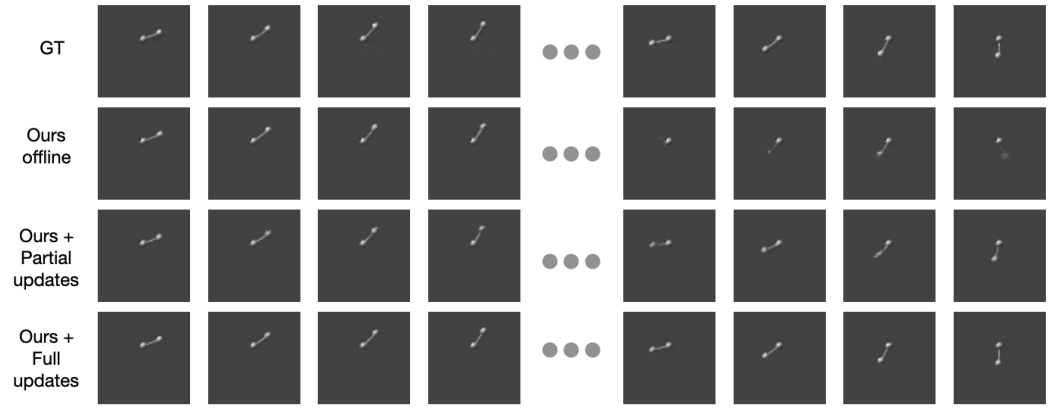

Figure 10: **Impact of online updates on the quality of the prediction for the pendulum with high oscillations amplitude (between 0 and 2π radians) on a dataset of pendulums of length varying between 0.3 m and 0.8 m.** The first row shows ground truth (GT) images. The second row shows predicted frames without updates. The third row shows predicted frames with our model trained with partial online updates (every 15 measurements). The last row shows predicted frames with online updates performed at each new measurement.

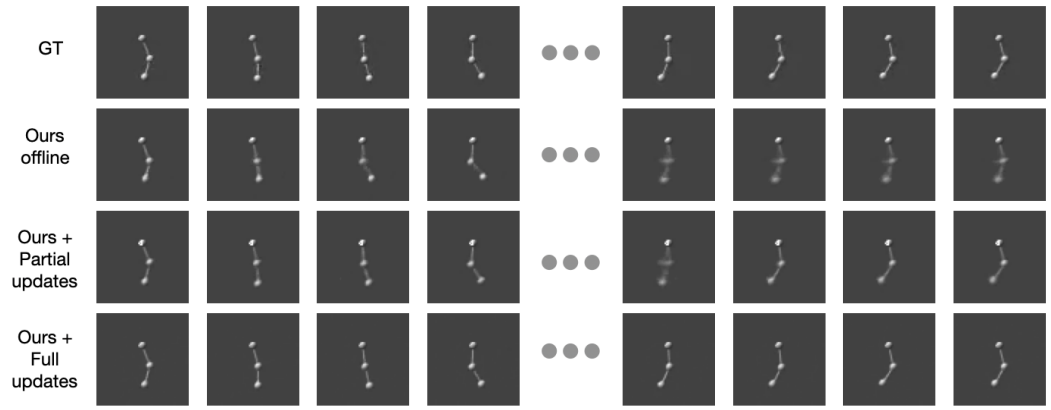

Figure 11: **Impact of online updates on the quality of the prediction for the double pendulum.** The first row shows ground truth (GT) images. The second row shows predicted frames without updates. The third row shows predicted frames with our model trained with partial online updates (every 15 measurements). The last row shows predicted frames with online updates performed at each new measurement.

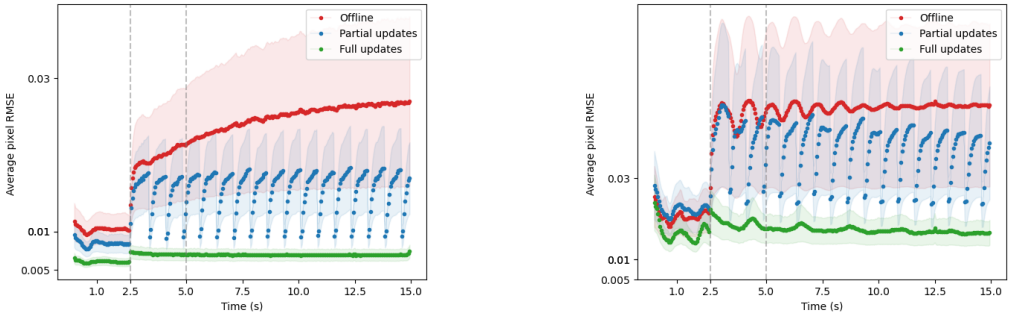

Figure 12: Average per-pixel RMSE loss over a 15s prediction horizon. **Left:** Pendulum with high amplitude oscillations (between 0 and 2π radians). **Right:** Double pendulum with a first pole oscillating between $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ radians.

Control.
Figure 13 shows the trajectory obtained when driving the cartpole from an initial state specified by two consecutive frames (red) to a position where the pole is inverted. [2] Solving the QP problem of Eq. (13) of the main submission returns a sequence of controls $[u_1, \ldots, u_{10}]$ that are applied starting from z_0 and z_1, the embeddings of the two first frames (red), such that for $0 \leq t \leq 11$:

$$z_{t+2} = A_1 z_t + A_2 z_{t+1} + B u_{t+1} \tag{16}$$

where A_1, A_2 are blocks of the matrix $\tilde{A}$ described in Eq. (10) of the main submission, in the case where $h = 2$. The frames are obtained by decoding the sequence $[z_0, \ldots, z_{11}]$.

Figure 13: **Illustration of cartpole control.** Starting from an initial state specified by two consecutive frames (in red), we estimate and apply the controls necessary to guide the pole to an inverted position in a horizon of 0.5 s. The last frame shows the final position of the cartpole after all controls were applied (in green).

A.3 Experiments on fluids

We extended our approach to the study of a fluid flowing past a cylinder following [6], through the study of four of its physical quantities: density, x-momentum, y-momentum and energy.

A.3.1 Dataset generation

We followed the dataset generation protocol described in [6]: the solver PyFR [53]is used to solve the Navier-Stokes equation [54] for each one of the four quantities mentioned in the previous paragraph,

[2] As described in the main submission, our prediction model uses not one but at least two codes in the latent space to predict the next one, which is why, for the control task, two initial frames are considered and used to constraint the QP problem described in Eq. (13) of the main submission.

with a discretization time step of 0.1 ms. The solutions are then formatted into 4-channels image-like inputs of size 128×256 (one channel per physical quantity), and one image is kept every 150 ms (every 1500 steps of the solver) for each of the four quantities. The simulation is run in two different settings: unforced and forced dynamics. In the unforced dynamics setting, the simulation is run during 636 seconds (which corresponds to a trajectory of 4328 frames) with no velocity being prescribed to the cylinder. In the forced dynamics setting, the simulation is run during 750 seconds, (which corresponds to a trajectory of 5000 frames) with a velocity being prescribed to the cylinder. The obtained trajectories are then split into respectively 1200 and 1600 overlapping sequences, both with a duration of 4.8 s.

A.3.2 Experimental protocol

We trained our model during 1000 epochs on 1200 32 frame-long (4.8 s) sequences in the case of unforced dynamics, and on 1600 32 frame-long (4.8 s) sequences in the case of forced dynamics. In this 32 frame-long sequence, encodings of the 16 first frames (2.4 s) were used to estimate the dynamics matrix A (and the control matrix B in the case of forced dynamics), and the 16 (2.4 s) following frames were predicted. We used a batch-size of 16 and a latent dimension $n_z = 8$ in the case of unforced dynamics, and $n_z = 32$ in the case of forced dynamics. We set our initial learning rate to $1e^{-3}$, and divided it by 2 every 100 epochs.

A.3.3 Results

Prediction. We evaluated our model on 100 frame-long (15 s) sequences. Figure 14 shows the average L1 loss over time over 120 test trajectories using the three variations of our approach we detailed in the main paper. Even though we see that variations of our model that include updates (at every time step starting the 16th time step in green, or at a single time step, the 40th, in blue) have lower error values that do not grow over time compared to our offline variation (where no update of the model is performed), the error values for all three variations remain very low and are invisible to the naked eye, as can be seen in Fig. 15.
In this work, we followed the experimental protocol described in [6] for comparison, however, we believe that future work should consider multiple trajectories from different fluids (with different physical parameters) instead of only one unique trajectory of one fluid, and that the sequences used for training should not overlap.

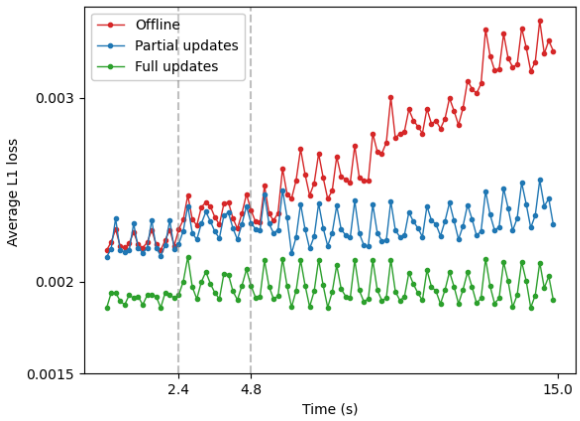

Figure 14: Average L1 loss on all four quantities of the fluid system over a 15 s prediction horizon.

Control.
Figure 16 shows the trajectory of the x-momentum of the studied fluid obtained when applying a sequence of controls to stabilize the fluid flow. The controls are a solution to the QP problem defined in equation (13) of the main submission where z_1 corresponds to an initial representation of the fluid, and z_f corresponds to a representation of the fluid where it is stabilized (i.e.; when its flow is laminar). Note that each code z_t is built by encoding the 4 physical quantities mentioned above at time t using

Figure 15: **Prediction of the x-momentum**. The top-left frame is the last frame of the 16 frame-long sequence that was used to build the dynamics matrix A. All the following frames are predicted.

the learned encoder, and that the resulting controlled sequence in Fig. 16 only shows one quantity (the x-momentum).

A.4 Training details

During training, we seek to minimize the loss defined in equation (12) of the main paper. For ease of reading, equation (12) only accounts for the case of unforced dynamics (i.e.; where the studied systems are not actuated, thus when we are only looking for the dynamics matrix A). In the case of actuated systems, an additional term is added to this loss such that it becomes:

$$\mathcal{L}_{\theta,\mu}(\{d_{1:T}\}_{i=1,\dots,N}) = \frac{1}{N}\sum_{i=1}^{N}\underbrace{\sum_{t=1}^{m}\|d_t^i - \Psi_\mu(\Phi_\theta(d_t^i))\|_2^2}_{\text{Auto-encoder loss}} + \underbrace{\sum_{t=m+1}^{T}\|d_t^i - \Psi_\mu(A_i^{t-m}\Phi_\theta(d_m^i) + B_i u_t)\|_2^2}_{\text{Prediction loss}}. \tag{17}$$

At each optimization step, A_i and B_i are estimated using equation (5) from the main paper for each trajectory i $[d_1^i, \dots, d_T^i]$. In fact, they are estimated from the first m codes of $[d_t^i]_t$ (obtained with the encoder Φ_θ), then used to predict future codes through the relation $z_{t+1}^i = A_i z_t^i + B_i u_t^i$. The sequence $[z_t^i]_t$ is then decoded using the decoder Ψ_μ. The parameters (θ, μ) of Φ_θ and Ψ_μ are then

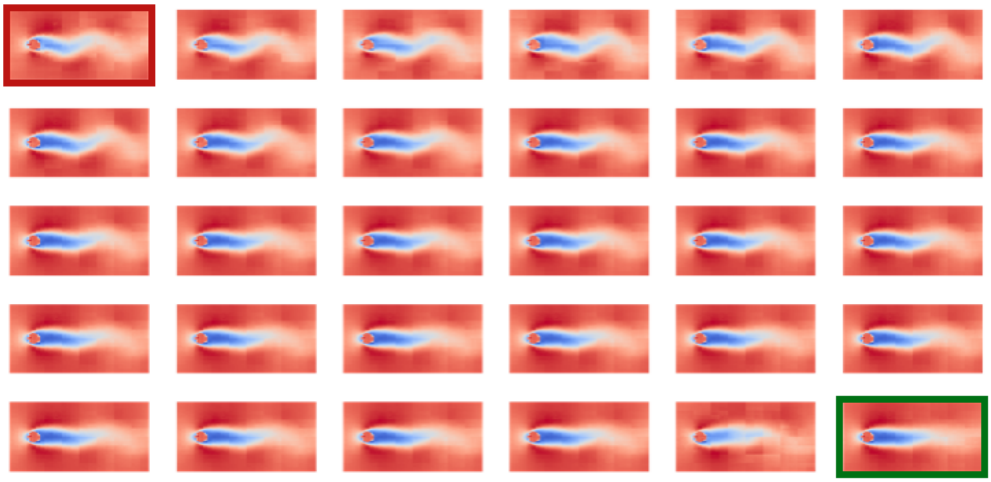

Figure 16: **Illustration of fluid control**. Starting from an initial configuration of the fluid at a given time step (top-left), we estimate and apply (in the learned latent space) the controls necessary to stabilize it (bottom-right). The quantity shown here is the x-momentum.

updated. In practice, we see that the term $\sum_{i,t} \| \| z_{t+1}^i - A_i z_t^i + B_i \| \|_2^2$ decreases during training without being explicitly minimized, as can be seen in Fig. 17.

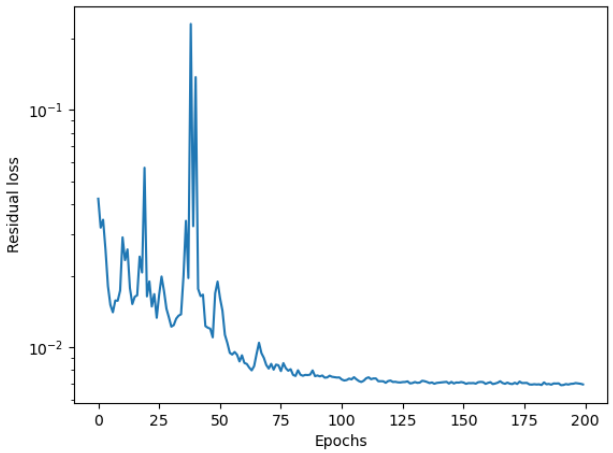

Figure 17: **Residual loss**. Evolution of $\sum_{i,t} \| \| z_{t+1}^i - A_i z_t^i + B_i \| \|_2^2$ during training.

B Online updates

The estimation of the matrix A requires inverting the matrix:

$$M = \begin{bmatrix} I_{n_z} & Z_1 \\ Z_1^T & -\rho I_{T-1} \end{bmatrix}. \tag{18}$$

This can be efficiently performed through a Cholesky decomposition of the form LDL^t because M is the KKT matrix associated to a saddle point problem, and has positive definite and negative definite blocks. In the case where our model is updated online, M must be recomputed at every update. We can avoid recomputing it from scratch by performing rank-1 updates of its Cholesky decomposition when new measurements are considered (which would correspond to adding one column to Z_1 and one row to Z_1^T).